Army Research Laboratory

Warriors Edge Simulation System

by Mark Thomas

ARL-TR-3430 February 2005

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-TR-3430 **February 2005**

Warriors Edge Simulation System

Mark Thomas
Computational and Information Sciences Directorate, ARL

Report Documentation Page	*Form Approved OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* February 2005	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)** January 2003–October 2003
4. TITLE AND SUBTITLE Warriors Edge Simulation System		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Mark Thomas		**5d. PROJECT NUMBER** P622783.Y10
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRD-ARL-CI-CT Aberdeen Proving Ground, MD 21005-5067		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-3430
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In August 2003, the U.S. Army Research Laboratory (ARL) participated in an Office of the Secretary of Defense technology insertion program named Horizontal Fusion (HF). The HF program is inserting cross-domain, web-based tools into the defense force to enhance situational awareness, decrease communications delays, and speed up the use of Internet protocols in the sharing of battlefield information. ARL's role in the HF program is the Army's story from the brigade to the platoon leader. ARL technology development for the HF program included individual soldier telecommunications and computer gear, platoon leader decision support tools to include visualization and message generation and distribution, a surrogate S2 brigade station, information fusion technology, and global web-space integration. The work period for this effort was 6 months. This report will briefly describe the Warriors Edge components and how simulation was used to stimulate, populate, and enhance the Warriors Edge demonstration.

15. SUBJECT TERMS
modeling and simulation, live virtual integration, Warriors Edge, horizontal fusion, dismounted infantry SAF

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Mark Thomas
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UL	28	**19b. TELEPHONE NUMBER** (*Include area code*) 410-278-5011

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

The contributions of many people went into the success of this effort. The author would like to thank the members of the U.S. Army Research Laboratory Distributed Simulation Team—Maria Lopez, Gary Moss, Andrew Neiderer, and Frederick Brundick, for code writing and creative solutions to the challenges of this project. The author also thanks Timothy Gregory and Robert Winkler of the Battlefield Visualization Branch for providing coding support for the Tactical Object System. Thanks also go to Dr. Phillip Emmerman, Dr. Windell Ingram, Pat Jones, and Barbara Broome for seeing the utility of modeling and simulation to the effort. The author also thanks Tim Hanratty and John Dumer for the S2 Workstation interfacing advice. Appreciation is given to Peter Gerken of Advanced Technology Laboratory for collaborating on the TACSUM interface.

INTENTIONALLY LEFT BLANK.

1. Introduction

In August 2003, the U.S. Army Research Laboratory (ARL) participated in an Office of the Secretary of Defense technology insertion program named Horizontal Fusion (HF). The HF program is inserting cross-domain, web-based tools into the defense force to enhance situational awareness, decrease communications delays, and speed up the use of Internet protocols (IP) in the sharing of battlefield information. ARL's role in the HF program is the Army's story from the brigade to the platoon leader. ARL technology development for the HF program included individual soldier telecommunications and computer gear, platoon leader decision support tools to include visualization and message generation and distribution, a surrogate S2 brigade station, information fusion technology, and global web-space integration. These systems had to be integrated, tested, and deployed in 6 months.

The scenario developed for the ARL effort was based on military operations in urban terrain operations. The scenario involved three platoons operating in and around a town. One platoon conducted surveillance and traffic control to the north, the second platoon conducted reconnaissance south of the first platoon, and the third platoon entered the town and executed operations there. Because of troop limitations, and to populate the global space with enough troops, simulation was used to add units, provide intelligence messages, and stimulate actions to be performed by the live troops.

This report will detail the simulation used in the ARL HF program Warriors Edge technology development and demonstration. The components of the Warriors Edge technology and the simulation software to exercise it will be described herein.

2. Warriors Edge Technology Components

Warriors Edge technology was developed to link the individual soldier to the global operational space. This was accomplished by developing leading edge technologies at the platoon and brigade levels around a common communications interface.

The platoon nodes each had modular universal laser equipment (MULE), robots, and individual soldier ensembles. All items were equipped with global positioning system devices, for tracking, and wireless communications. The MULE consisted of a vehicle and payload. The payload contained electrical power, computers, and multiple wireless communications networks. It served as the hub for the platoon communications system. There were three iRobot ATRV2s per platoon and three iRobot PacBots per platoon. Each ATRV2 and PacBot carried onboard sensors and cameras to relay information back to the MULE. The individual soldier ensembles were composed of computers for information processing, instrumented rifles, and voice-over IP communications.

The brigade node consisted of a prototype S2 analyst workstation developed by ARL. The S2 workstation provided a graphical interface to display troop positions and text messages. The S2 workstation also contained a web browser to connect to the HF web portal for information prompts and alerts, and provided screenshots of the current battle situation to the collateral space for viewing by interested parties. Figure 1 shows the Warriors Edge platoon components, and figure 2 shows the brigade components.

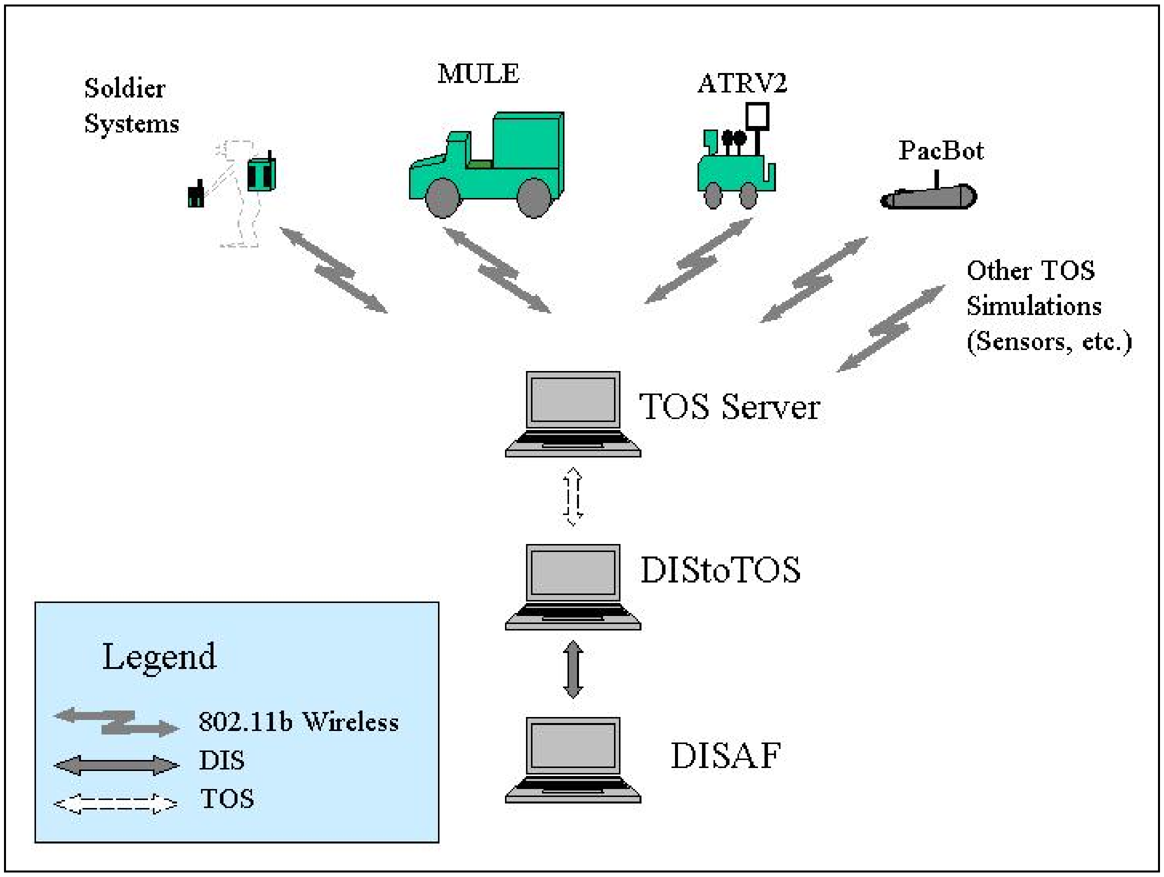

Figure 1. Platoon components.

3. Warriors Edge Simulation

Simulation was used to stimulate and populate the battle space. The simulation consisted of the Dismounted Infantry Semi-Automated Forces (DISAF) variant of the OneSAF Testbed, and ARL-developed network interface, data conversion, and text message transmission software.

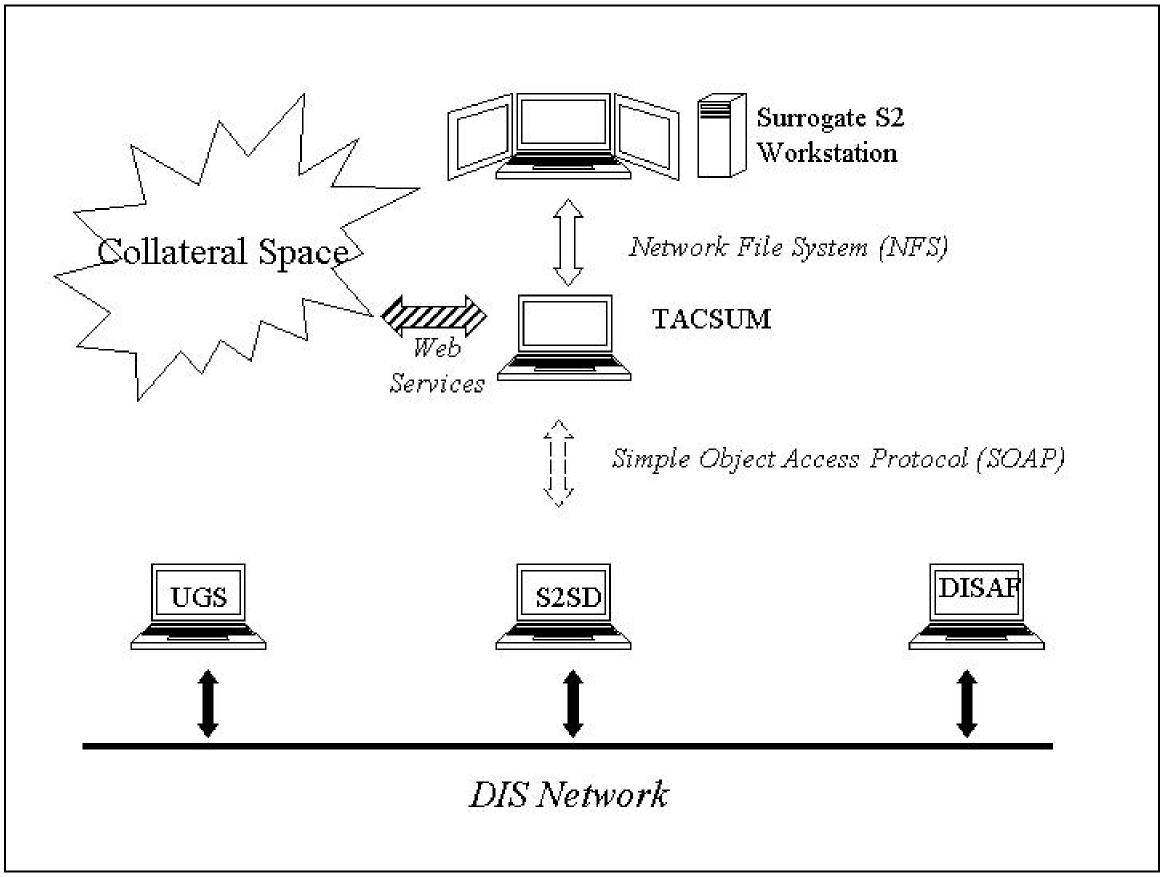

Figure 2. Brigade-level Warriors Edge components.

The role of simulation was two-fold. First, it would provide the necessary numbers of troops on the battlefield and represent the actions of troops and vehicles. Secondly, it would provide stimulation via simulated human intelligence messages to the S2 fusion station for intelligence gathering and scenario alerts. To accomplish this mission, a dual system of SAF and manual message transmission was used.

Multiple data formats were required. The SAF systems communicated using Distributed Interactive Simulation (DIS) data packets. The individual soldier, robots, and MULE communicated using the ARL Tactical Object System (TOS) protocol. The S2 station used an in-house XML data format for individual and unit position information.

DISAF

The DISAF was used to create and orchestrate the movements and actions of units on the battlefield. For Warriors Edge, there were three platoons in play. Two platoons were completely simulated by the DISAF. The third platoon was composed of one live squad and two simulated squads. Entities utilized in the simulation included friendly dismounted infantry fire teams, surrogate robots, unmanned infrared and acoustic ground sensors, and an enemy dismounted platoon. The surrogate robots were modeled using the unmanned ground sensor entity.

4. ARL-Developed Software

Two message translation programs were developed. One served the platoon and below echelons, and the other served above platoon. The platoon-level program, DIStoTOS (DtoT), was designed to communicate between the individual soldier ensembles, robots, MULE, and the DISAF. DtoT used the TOS communications protocol developed by ARL combined with a DIS interface to communicate between DISAF entities and live entities on the battlefield. S2StationDriver (S2SD) converted simulation data on the DIS network into XML formatted strings for use by the Surrogate S2 Station via a web interface. In addition, S2SD provided the user with the capability to transmit XML files to the web server as needed. These programs will be described below.

4.1 DIStoTOS

DtoT runs in real time. It provides bidirectional message flow between the DIS network and the TOS. The program listens to the DIS network, keeps a database of entities on the DIS network, and formats TOS messages to create, update, and delete entities as necessary. It is intended to populate a live exercise with simulated entities and to send live entity posture data to the simulation network.

Information sent to TOS from the DIS network included entity identification, latitude and longitude, entity type, and parent unit. Entity types were expressed as MIL-STD-2525B[1] icon strings. Icon strings were implemented for friendly and enemy dismounted infantry, the MULE, a High Mobility Multipurpose Wheeled Vehicle (HMMWV), the unmanned guided vehicle (UGV), a generic enemy vehicle, friendly unmanned ground robots, and an unknown entity. Because DIS uses enumerated data for entity types, and TOS uses MIL-STD-2525B icon strings, a mapping between DIS entity enumerations and MIL-STD-2525B icons was implemented as follows (table 1):

Table 1. Mapping from DIS enumerated type to MIL-STD-2525B icon type.

DIS Kind	DIS Country	DIS Category	DIS Subcategory	MIL-STD-2525B Icon
3	225	NA	NA	FriendlyDI
3	222	NA	NA	EnemyDI
1	225	6	2	MULE
1	225	6	1	HMMWV
1	225	27	NA	UGV
1	222	NA	NA	Enemy vehicle

Note: NA = not applicable.

[1]MIL-STD-2525B. *Common Warfighting Symbology* **1999**.

Mapping from TOS icons to DIS entity enumerations was implemented as follows (table 2):

Table 2. Mapping from MIL-STD-2525B icon string to DIS enumerated type.

Object/MIL STD 2525 Icon String	DIS Kind	DIS Domain	DIS Country	DIS Category	DIS Subcategory
EnemyDI "SHGPEWR--------"	3	1	222	1	205
FriendlyDI "SFGPEWR--------"	3	1	225	1	32
MULE "SFGPEVU--------"	1	1	225	6	2
HMMWV "SFGPEVUX-------"	1	1	225	6	1
PacBot "SEPGEVAL-------"	1	1	225	27	2
ATRV2 "SEPGVAL--------"	1	1	225	27	2
UGS "SFGPES---------"	1	1	225	27	2
Enemy Vehicle "SHGPEVUX-------"	1	1	222	2	3

DIS entity information includes an 11-character marking field. This field was used to identify each individual entity as well as indicate force structure. This mapping was implemented as follows (table 3):

Table 3. Conversion from DIS entity marking to platoon.

Entity Type/Marking Field Position	5	6	Result
Dismounted infantry	1, 2, or 3	NA	Platoon 1, platoon 2, platoon 3
All others	5, 6, 7, 8, 9	1, 2, or 3	Platoon 1, platoon 2, platoon 3
All others	NA	NA	Unknown

Note: NA = not applicable.

For example, a marking of "100A264" is a member of platoon 2 if the entity is a dismounted infantryman. A marking of "100A521" is a member of platoon 2 if the entity is a vehicle.

Coordinate conversion from latitude/longitude to DIS world coordinates was accomplished using the Synthetic Environment Data Representation Interface Standard coordinate conversion routines. Inputs to the converter were the Universal Transverse Mercator Easting and Northing reference point and the grid zone number.

An entity list was kept for the DIS and the TOS networks. Because of the DIS entity timeout in DISAF, TOS entity information needed frequent updates whether TOS updates or entity movement occurred or not. A memory resident database of TOS entities was kept, and their data were transmitted to the DIS network as Entity State PDUs every 5 s.

The TOS network required no heartbeat. DIS entity information was sent to the TOS network when an entity was discovered, if it moved, once/min, or when it was deleted. Frequent updating of the TOS network caused latency problems in the TOS system, so it was decided to only update DIS information to the TOS network as required.

TOS network data are XML formatted. The Xerces parser from xml.apache.org was used to perform the parsing from XML into the entity list structure.

The DtoT has a command line interface to list entities, change the DIS entity timeout, delete all TOS entities owned by DtoT, add DIS entities in DtoT to the TOS network, and shutdown the program.

4.2 S2StationDriver

S2SD provided data to the prototype S2 Station in the Warriors Edge brigade-level web-based information portal. Data provided were used to track friendly entities and insert messages used for data fusion and information dissemination. S2SD provided a link between the web-based prototype S2 Station and the DIS simulation network.

The prototype S2 workstation used a webserver for web interactions. The webserver, called the TACSUM, used an XML data format for all web interactions. S2SD communicated to the TACSUM, which then forwarded the message to the prototype S2 workstation.

An XML specification for TACSUM web interactions was developed. The specification, called *dcgsa*, provided messages to convey posture data, freetext messages, and other battlefield information. The *dcgsa* schema included data from the *midb*, *gmi*, and *gov* XML message specifications for comprehensive message content.

Data flow between the S2SD and the TACSUM was via the Simple Object Access Protocol. When data were ready for uploading to the TACSUM, a socket was created between the S2SD and TACSUM, the data sent, and the socket closed.

S2SD aggregated DIS entity data into units for the S2 station. The S2 station, being a brigade-level asset, displayed both individual entities and units. For this exercise, individual entities were aggregated according to the process described in table 1. Two files were created. One, the *individual_dcgsa.xml* file, was used to send posture data for each individual friendly entity in the simulation. The second, *units_dcgsa.xml*, sent unit data (platoon 1, platoon 2, and platoon 3). Each file was sent per iteration. On the first iteration, all entities were sent using the *individual_dcgsa.xml* file. After that, only entities that moved were sent via the *individual_dcgsa.xml* file. The *units.xml* file was sent each iteration.

Unit location was computed as the average location of all entities in the unit. There was a choice between using the platoon leader's location or the average, and the average was chosen. Because unit location changes slowly, this fit well with the S2 workstation's update rate of once every 2 min.

In addition to the automatic posture data updates, the S2SD provides the user the ability to send an XML-formatted file to the TACSUM manually. This capability gives the S2SD the flexibility to simulate multiple data inputs (human and signal intelligence) to the TACSUM for information fusion. For the HF demonstration, the S2SD's XML message transmission capability was used to initiate scenario events by sending messages at scripted times.

The *individual_dcgsa.xml* file follows the following form:

```
<?xml version='1.0' encoding='UTF-8'?>
<SIM_Update
   xmlns='simInterface.xsd'
   xmlns:sim='simInterface.xsd'
   xmlns:GMI='urn:midb:gmi:gov'
   xmlns:dcgsa='http://www.pmif.army.mil/DCGSA'
   xmlns:mil='urn:midb:gmi:gov'>
 <simUpdate>
 <dcgsa:DATETIME_CREATED>2003-07-
18T19:38:33.000</dcgsa:DATETIME_CREATED>
 <dcgsa:IND>
    <dcgsa:PARENT_UNIT_IDENTIFICATION>Platoon
3</dcgsa:PARENT_UNIT_IDENTIFICATION>
    <dcgsa:INDIVIDUAL_MASTER_KEY>100A351</dcgsa:INDIVIDUAL_MASTER_KEY>
    <GMI:AFFILIATION>F</GMI:AFFILIATION>
   <dcgsa:IND_ADDRESS>
    <dcgsa:INDIVIDUAL_MASTER_KEY>100A351</dcgsa:INDIVIDUAL_MASTER_KEY>
    <dcgsa:INDIVIDUAL_SEQUENCE>1</dcgsa:INDIVIDUAL_SEQUENCE>
    <dcgsa:LATITUDE>32.385556</dcgsa:LATITUDE>
    <dcgsa:LONGITUDE>-84.792931</dcgsa:LONGITUDE>
    <GMI:ELEVATION>0.010143</GMI:ELEVATION>
   </dcgsa:IND_ADDRESS>
 </dcgsa:IND>
 <dcgsa:IND>
    <dcgsa:PARENT_UNIT_IDENTIFICATION>Platoon
1</dcgsa:PARENT_UNIT_IDENTIFICATION>
    <dcgsa:INDIVIDUAL_MASTER_KEY>100A112</dcgsa:INDIVIDUAL_MASTER_KEY>
    <GMI:AFFILIATION>F</GMI:AFFILIATION>
   <dcgsa:IND_ADDRESS>
    <dcgsa:INDIVIDUAL_MASTER_KEY>100A112</dcgsa:INDIVIDUAL_MASTER_KEY>
    <dcgsa:INDIVIDUAL_SEQUENCE>1</dcgsa:INDIVIDUAL_SEQUENCE>
    <dcgsa:LATITUDE>32.373058</dcgsa:LATITUDE>
    <dcgsa:LONGITUDE>-84.794886</dcgsa:LONGITUDE>
    <GMI:ELEVATION>116.041748</GMI:ELEVATION>
   </dcgsa:IND_ADDRESS>
 </dcgsa:IND>
 </simUpdate>
</SIM_Update>
```

The *unit_dcgsa.xml* file has the following format:

```
<?xml version='1.0' encoding='UTF-8'?>
<SIM_Update
```

```
    xmlns='simInterface.xsd'
    xmlns:sim='simInterface.xsd'
    xmlns:GMI='urn:midb:gmi:gov'
    xmlns:dcgsa='http://www.pmif.army.mil/DCGSA'
    xmlns:mil='urn:midb:gmi:gov'>
  <simUpdate>
  <dcgsa:DATETIME_CREATED>2003-07-
18T19:38:33.000</dcgsa:DATETIME_CREATED>
  <dcgsa:UNIT>
    <dcgsa:TRUE_UNIT_DESIGNATION>Platoon 1</dcgsa:TRUE_UNIT_DESIGNATION>
    <dcgsa:FUNCT_ROLE>541</dcgsa:FUNCT_ROLE>
    <dcgsa:INTEL_PEDIGREE>1</dcgsa:INTEL_PEDIGREE>
      <dcgsa:UNIT_MASTER_KEY>Platoon 1</dcgsa:UNIT_MASTER_KEY>
    <dcgsa:LATITUDE>32.371780</dcgsa:LATITUDE>
    <dcgsa:LONGITUDE>-84.799443</dcgsa:LONGITUDE>
  </dcgsa:UNIT>
  <dcgsa:UNIT>
    <dcgsa:TRUE_UNIT_DESIGNATION>Platoon 2</dcgsa:TRUE_UNIT_DESIGNATION>
    <dcgsa:FUNCT_ROLE>541</dcgsa:FUNCT_ROLE>
    <dcgsa:INTEL_PEDIGREE>1</dcgsa:INTEL_PEDIGREE>
      <dcgsa:UNIT_MASTER_KEY>Platoon 2</dcgsa:UNIT_MASTER_KEY>
    <dcgsa:LATITUDE>32.371713</dcgsa:LATITUDE>
    <dcgsa:LONGITUDE>-84.809940</dcgsa:LONGITUDE>
  </dcgsa:UNIT>
  <dcgsa:UNIT>
    <dcgsa:TRUE_UNIT_DESIGNATION>Platoon 3</dcgsa:TRUE_UNIT_DESIGNATION>
    <dcgsa:FUNCT_ROLE>541</dcgsa:FUNCT_ROLE>
    <dcgsa:INTEL_PEDIGREE>1</dcgsa:INTEL_PEDIGREE>
      <dcgsa:UNIT_MASTER_KEY>Platoon 3</dcgsa:UNIT_MASTER_KEY>
    <dcgsa:LATITUDE>32.385605</dcgsa:LATITUDE>
    <dcgsa:LONGITUDE>-84.793055</dcgsa:LONGITUDE>
  </dcgsa:UNIT>
  </simUpdate>
</SIM_Update>
```

Transmitting the XML files was performed using the POST method. The function, PostFileToWebServer, takes three arguments. The first argument is the hostname of the webserver. The second argument is the name of the file to post. This must be the entire pathname of the file. The third argument is the port that the webserver is listening to for connections. PostFileToWebServer calls helper functions to open a connection, read in the file, prepend and append appropriate headers and footers, and send the file. After sending the file, PostFileToWebServer() performs a select() on the socket to detect a response from the webserver(not used). The socket is then closed, and the function returns. The implementation of the web posting is included in the appendix.

5. Conclusion

An exercise to highlight the force multiplication from the use of web-based technologies was conducted. The U.S. Army component, called Warriors Edge, was a vital element to the success of the exercise. To demonstrate the Army system, software was written to web-enable certain Army functions. The merger of live and simulated Warriors Edge components and systems provided a realistic, exciting exercise and demonstration of the power of web-enabled systems to mission effectiveness, increased soldier survivability, and lethality.

Future directions for this software development and research are in the areas of fusion modeling and simulation, human intelligence inference from scenario events, and increased sensor types. In addition, a more comprehensive graphical user interface will provide more capability to the user. For example, given the capability to type freetext messages, a user can inject any type of message from any sensor desired.

INTENTIONALLY LEFT BLANK.

Appendix. POST Web Method Source Code

The following code is the implementation of the POST web method. The POST method is used to send data to a web server. In the S2StationDriver, this function was used to send the *individual_dcgsa.xml* and *unit_dcgsa.xml* files to the TACSUM web server.

```
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <sys/time.h>
#include <fcntl.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <iostream.h>

#include <fstream.h>
//
// This code uploads a file or files to a web
// server
//

struct sockaddr_in server;
static int fd;          // The server socket to write to
int updateRate = 5;
int fileCounter = 0;
//
//
//
int OpenConnection( char *host, int port )
{
  struct hostent *hp = gethostbyname( host );

  if( hp == NULL ){
    cerr << "ERROR - OpenConnection: hostname not found" << endl;
    return( 0 );
  }

  bzero( (char *)&server, sizeof( server ) );
  bcopy( hp->h_addr, (char *)&server.sin_addr, hp->h_length );
  server.sin_family = hp->h_addrtype;
  server.sin_port = htons( port );

  fd = socket( AF_INET, SOCK_STREAM, 0 );
  if( fd < 0 ){
    cerr << "OpenConnection : socket call failed" << endl;
    return( 0 );
  }

  int i;

  if( connect( fd, (struct sockaddr *)&server, sizeof( server )) < 0 ){
    cerr << strerror( errno ) << endl;
    return( 0 );
  }

  cerr << "Connection Established" << endl;
  return( 1 );
}
//
//
//
static int getSize( char *filename )
{
  struct stat buf;
  if( stat( filename, &buf ) == 0 )
    return( buf.st_size );
  else{
    cerr << "ERROR - stat of " << filename << " unsuccessful" << endl;
    return( 0 );
  }
}
//
//
//
char *importFile( char *filename )
{
  FILE *fp = fopen( filename, "r" );
  if( fp == NULL )
    return( NULL );
  int fileSize = getSize( filename );
  char *p = (char *)malloc( fileSize + 1);
  if( p == NULL ){
    cerr << "importFile: Cannot malloc " << fileSize << " bytes" << endl;
    return( NULL );
  }

  char str[ 128 ];

  int first = 1;
  int count = 0;

  while( fgets( str, 128, fp ) != NULL ){
    if( first ){
      strcpy( p, str );
      first = 0;
    }
    else
      strcat( p, str );

    count += strlen( str );
  }

  fclose( fp );

  return( p );
}

static char *postHeader = "POST /dcgsa/servlet/EntityStatusServices HTTP/1.0\nFrom:
WarriorsEdge\nUser-Agent: WarriorsEdgeSim\nContent-Type: text/xml";
//
//
//

void PostFile( char *filename )
{
  extern int recordFiles;
  extern int playFiles;
  extern char archiveDir[];
  extern ifstream indexIn;
  extern ofstream indexOut;
  extern time_t missionStart;

  time_t timeNow;
  time_t missionTime;
  struct tm *gmStruct;
  char   timeStamp[32];
  char   *timeBuf;

  int sent;

  char *bigBuffer;

  char *fileData = importFile( filename );

  if( fileData == NULL ){
    cerr << "ERROR - PostFile: cannot open file " << filename << endl;
    return;
  }

  timeNow = time(0);
  missionTime = timeNow - (time_t)missionStart;

// update time stamp in XML file
// if you want to clobber the date but not the time,
//  simply make the timeStamp string shorter
  gmStruct = gmtime(&timeNow);
  sprintf(timeStamp, "%4d-%02d-%02dT%02d:%02d:%02d",
    gmStruct->tm_year + 1900,
    gmStruct->tm_mon + 1,
    gmStruct->tm_mday,
    gmStruct->tm_hour,
    gmStruct->tm_min,
    gmStruct->tm_sec);

// strlen("DATETIME_CREATED>") == 17
// strlen(timeStamp) == 19  ("2003-06-12Z17:25:20")
  timeBuf = strstr(fileData, "DATETIME_CREATED>");

  if( timeBuf == NULL ){
    cerr << "No Datetime in this message" << endl;
  }
  else{
    timeBuf += 17;

    cerr << "Replacing datetime " << timeBuf << endl;

    for (int i = 0; i < 19; i++)
      *timeBuf++ = timeStamp[i];
  }
// copy data to archive file (if desired)
  if (recordFiles){
    int copyFd;
    char newName[128];

    ++fileCounter;
    sprintf(newName, "%s/%05d%s", archiveDir, fileCounter, filename);
    copyFd = open(newName, O_CREAT | O_WRONLY, 0777);
    sent = write( copyFd, fileData, strlen( fileData ));

    if( sent < strlen( fileData) ){
      cerr << "PostFile : short write on file copy" << endl;
    }
    else{
      cerr << "copied " << filename << " to " << newName << endl;
    }
    close(copyFd);

// save log info in index file
    indexOut << missionTime << ' ' << newName << endl;
    indexOut.flush();
  }

  char headerD[128];
  sprintf( headerD,"%s\n",filename );
  int contentLength = strlen( fileData ) + strlen( headerD );

  bigBuffer = (char *)malloc( contentLength + 1024 );

  sprintf( bigBuffer,"%s\nContent-Length:%d\n\n%s\n%s\n\0",
           postHeader, contentLength, filename, fileData );

  sent = write( fd, bigBuffer, strlen( bigBuffer ));

  if( sent < strlen( bigBuffer) ){
    cerr << "PostFile : short write" << endl;
  }

  free( fileData );
  free( bigBuffer );
}
//
//
//
//
int PostFileToWebServer( char *serverName, char *fileName, int port )
{
  struct timeval timeout;
  char response[64000];

  cout << "Posting " << fileName << " to " << serverName << " on port " << port << endl;

  if( !OpenConnection( serverName, port ) ){
    cerr << "ERROR - cannot open connection to web server " << endl;
    return( 0 );
  }

  PostFile( fileName );

  int ready;

  fd_set readSet;

  FD_ZERO( &readSet );
  FD_SET( fd, &readSet );

  timeout.tv_sec = updateRate;
  timeout.tv_usec = 0;

  ready = select( fd + 1, &readSet, 0, 0, &timeout );

  close( fd );

  return( 1 );
}
```

Bibliography

The Apache Software Foundation. Xerces C++ Parser. http://www.apache.org/xerces-c/index.html, 2002–2003.

Harold, E.; Means, W. *XML In A Nutshell*, 2nd Ed.; O'Reilly and Associates, Inc.: Sebastopol, CA, 2002.

Institute for Simulation and Training. *Enumeration and Bit-Encoded Values for use With IEEE 1278.1-1994, Distributed Interactive Simulation – Application Protocols*; IST-CR-93-46; Institute for Simulation and Training: Orlando, FL, 1993.

Lockheed Martin Advanced Distributed Simulation. *Users Manual for OTB SAF V1.0*. On-line manual, 1999.

NO. OF COPIES	ORGANIZATION
1 (PDF ONLY)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN STE 400 AUSTIN TX 78759-5316
1	US MILITARY ACADEMY MATH SCI CTR EXCELLENCE MADN MATH THAYER HALL WEST POINT NY 10996-1786
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CS IS T 2800 POWDER MILL RD ADELPHI MD 20783-1197

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES	ORGANIZATION
1	US ARMY SIMULATION TRNG AND INSTRNTL COM J STAHL 12350 RESRCH PRKWY ORLANDO FL 32826-3736
1	US SBCCOM NATICK SOLDIER CTR AMSSB RSS MA (N) D TUCKER KANSAS ST NATICK MA 01760-5020
1	US ARMY RSRCH INST B KNERR 12350 RSRCH PRKWY ORLANDO, FL 32826-3276
1	US ARMY RSRCH DVPMNT AND ENG COM J GROSSE 12423 RSRCH PKWY ORLANDO FL 32828
2	USARL AMSRD ARL CI CB L TOKARCIK R WINKLER 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	USARL AMSRD ARL CI J GOWENS 2800 POWDER MILL RD ADELPHI MD 20783-1197

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
15	DIR USARL AMSRD ARL CI CT M THOMAS (6 CPS) F BRUNDICK M LOPEZ G MOSS P JONES AMSRD ARL SL BB R SANDMEYER P TANENBAUM AMSRD ARL SL BE L BUTLER C KENNEDY AMSRD ARL WM BF G SAUERBORN

INTENTIONALLY LEFT BLANK.